1. ChangeDetectionStrategy?
2. Angular 2 security

* Angular by default considers all values as untrusted and satinizes all values before putting value into dom.
* To bind application data containing html and to render that html, use [innerHtml] instead of interpolation {{htmlsnippet}}.
* Avoid script tag.
* Avoid generating dynamic template.
* To mark a value as trusted, inject [DomSanitizer](https://angular.io/api/platform-browser/DomSanitizer).
* In a common anti-XSRF technique, the application server sends a randomly generated authentication token in a cookie. The client code reads the cookie and adds a custom request header with the token in all subsequent requests. The server compares the received cookie value to the request header value and rejects the request if the values are missing or don't match.

1. AOT compiler

* Ahead of time compiler
* Compiler at build time

1. Different types of metadata – host?
2. Pagination?
3. <https://stackoverflow.com/questions/38236313/angular2-inject-external-component-into-other-component-via-directive-not-worki>
4. How to inject another module component into another module’s component?

Try to export it

@NgModule({

declarations: [TaskCardComponent],

imports: [MdCardModule],

exports: [TaskCardComponent] <== this line

})

export class TaskModule{}

<https://stackoverflow.com/questions/39601784/angular-2-use-component-from-another-module>

1. How to use one’s component properties into another component
2. Metadata tells Angular how to process a class. It's not a component until you tell Angular about it. To tell Angular that HeroListComponent is a component, attach **metadata** to the class. In TypeScript, you attach metadata by using a **decorator**.
3. In two-way binding, a data property value flows to the input box from the component as with property binding. The user's changes also flow back to the component.
4. Apply conditional class

<div [class.active]=”row === 2”></div>

**Structural** directives alter layout by adding, removing, and replacing elements in DOM. (\*ngFor, \*ngIf)

1. The backtick (`)—which is not the same character as a single quote (')—allows you to compose a string over several lines.
2. **Attribute** directives alter the appearance or behavior of an existing element. (\*ngModel)
3. With \*ngIf, Angular isn't showing and hiding the message. It is adding and removing the paragraph element from the DOM.
4. Template reference variable?
5. Template expressions cannot refer to anything in the global namespace. They can't refer to window or document. They can't call console.log or Math.max.
6. <button [style.color]="isSpecial ? 'red' : 'green'">
7. <img [src]="heroImageUrl"> or <img bind-src="heroImageUrl">
8. Property binding or interpolation

<p><img src="{{heroImageUrl}}"> is the <i>interpolated</i> image.</p> <p><img [src]="heroImageUrl"> is the <i>property bound</i> image.</p>

1. <button [style.color]="isSpecial ? 'red': 'green'">Red</button>
2. <button [style.background-color]="canSave ? 'cyan': 'grey'" >Save</button>
3. <button on-click="onSave()">On Save</button>
4. \*ngFor with trackBy

With no trackBy, both buttons trigger complete DOM element replacement.

With trackBy, only changing the id triggers element replacement.

1. ngSwitch

<div [*ngSwitch*]="testNgSwitch">

<div \**ngSwitchCase*="'1'">Displayed 1 as per ngswitch condition</div>

<div \**ngSwitchCase*="'2'">Displayed 2 as per ngswitch condition</div>

</div>

**Bind to**[ngSwitch]. You'll get an error if you try to set \*ngSwitch because [NgSwitch](https://angular.io/api/common/NgSwitch) is an attribute directive, not a structural directive. It changes the behavior of its companion directives. It doesn't touch the DOM directly.

**Bind to**\*ngSwitchCase**and**\*ngSwitchDefault. The [NgSwitchCase](https://angular.io/api/common/NgSwitchCase) and [NgSwitchDefault](https://angular.io/api/common/NgSwitchDefault) directives are structural directives because they add or remove elements from the DOM.

1. <input #phone placeholder="phone number">

<!-- lots of other elements -->

<!-- phone refers to the input element; pass its `value` to an event handler <button (click)="callPhone(phone.value)">Call</button>

1. <form (ngSubmit)="onSubmit(heroForm)" #heroForm="ngForm">

<div class="form-group">

<label for="name">

Name <input class="form-control" name="name" required [(ngModel)]="hero.name"> </label> </div> <button type="submit" [disabled]="!heroForm.form.valid">Submit</button> </form>

1. The Angular **safe navigation operator (**?.**)** is a fluent and convenient way to guard against null and undefined values in property paths.

The current hero's name is {{currentHero?.name}}

1. **The non-null assertion operator ( ! ) -** disallow null and undefined by default. The type checker throws an error if you leave a variable unassigned or try to assign null or undefined to a variable whose type disallows null and undefined.
2. @Component({

selector: 'my-child-view',

template: '<input [(ngModel)]="hero">'

})

export class ChildViewComponent {

hero = 'Magneta';

}

template: `

<div>-- child view begins --</div>

<my-child-view></my-child-view>

<div>-- child view ends --</div>`

export class AfterViewComponent implements AfterViewChecked, AfterViewInit {

private prevHero = '';

@ViewChild(ChildViewComponent) viewChild: ChildViewComponent;

}

1. Difference between AfterView and AfterContent? - <https://scotch.io/tutorials/angular-2-transclusion-using-ng-content>
2. ngOnDestroy()

{

// prevent memory leak when

this.subscription.unsubscribe();

}

1. Use the :host pseudo-class selector to target styles in the element that hosts the component (as opposed to targeting elements inside the component's template).

:host { display: block; border: 1px solid black; }

1. use [ComponentFactoryResolver](https://angular.io/api/core/ComponentFactoryResolver) to add components dynamically.
2. Apply animations
3. <input #box (keyup.enter)="onEnter(box.value)">
4. The alterEgo is optional, so the constructor lets you omit it; note the question mark (?) in alterEgo?.
5. Apply form validation min/max length

<input type="text" id="name" class="form-control" required minlength="4" maxlength="24" name="name" [(ngModel)]="hero.name" #name="ngModel" >

<div [hidden]="!name.errors.minlength"> Name must be at least 4 characters long. </div>

1. **Only**declarables — components, directives and pipes — belong in the declarations array. Do not put any other kind of class in declarations; not [NgModule](https://angular.io/api/core/NgModule) classes, not service classes, not model classes.
2. [BrowserModule](https://angular.io/api/platform-browser/BrowserModule) registers critical application service providers. It also includes common directives like [NgIf](https://angular.io/api/common/NgIf) and [NgFor](https://angular.io/api/common/NgFor), which become immediately visible and usable in any of this module's component templates.
3. [BrowserModule](https://angular.io/api/platform-browser/BrowserModule) provides services that are essential to launch and run a browser app. [BrowserModule](https://angular.io/api/platform-browser/BrowserModule) also re-exports [CommonModule](https://angular.io/api/common/CommonModule) from @angular/common, which means that components in the AppModule module also have access to the Angular directives every app needs, such as [NgIf](https://angular.io/api/common/NgIf) and [NgFor](https://angular.io/api/common/NgFor).

Do not import [BrowserModule](https://angular.io/api/platform-browser/BrowserModule) in any other module. Feature modules and lazy-loaded modules should import [CommonModule](https://angular.io/api/common/CommonModule) instead.

1. entryComponents array is used to define ONLY component that are not found in html and created dynamically with ComponentFactoryResolver.